Non-Parametric Methods and Support Vector Machines

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Machine Learning
Outline

1 Non-Parametric Methods
   - $K$-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Outline

1 Non-Parametric Methods
   - $K$-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Outline

1 Non-Parametric Methods
   - $K$-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
The $K$-nearest neighbor ($K$-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $x$:

- Choose the number $K$ and a distance metric.
- Find the $K$ nearest neighbors of a given point $x$.
- Predict the label of $x$ by the majority vote (in classification) or average (in regression) of NNs' labels.
The *K*-nearest neighbor (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $x$:

1. Choose the number $K$ and a distance metric.
The $K$-nearest neighbor ($K$-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $x$:

1. Choose the number $K$ and a distance metric
2. Find the $K$ nearest neighbors of a given point $x$
$K$-NN Methods I

The *$K$-nearest neighbor* ($K$-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\mathbf{x}$:

1. Choose the number $K$ and a distance metric
2. Find the $K$ nearest neighbors of a given point $\mathbf{x}$
3. Predict the label of $\mathbf{x}$ by the majority vote (in classification) or average (in regression) of NNs’ labels

Distance metric?
E.g., Euclidean distance

Training algorithm?
Simply “remember” $\mathbf{X}$ in storage
The **K-nearest neighbor** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $x$:

1. Choose the number $K$ and a distance metric
2. Find the $K$ nearest neighbors of a given point $x$
3. Predict the label of $x$ by the majority vote (in classification) or average (in regression) of NNs’ labels

**Distance metric?**
K-NN Methods I

The **K-nearest neighbor** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $x$:

1. Choose the number $K$ and a distance metric
2. Find the $K$ nearest neighbors of a given point $x$
3. Predict the label of $x$ by the majority vote (in classification) or average (in regression) of NNs’ labels

- Distance metric? E.g., Euclidean distance $d(x^{(i)}, x) = \|x^{(i)} - x\|$
- Training algorithm?
The $K$-nearest neighbor ($K$-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $x$:

1. Choose the number $K$ and a distance metric
2. Find the $K$ nearest neighbors of a given point $x$
3. Predict the label of $x$ by the majority vote (in classification) or average (in regression) of NNs’ labels

- Distance metric? E.g., Euclidean distance $d(x^{(i)}, x) = \|x^{(i)} - x\|$
- Training algorithm? Simply “remember” $X$ in storage
**K-NN Methods II**

- Could be very complex
- \( K \) is a hyperparameter controlling the model complexity
Non-Parametric Methods

- \( K \)-NN method is a special case of non-parametric (or memory-based) methods.
Non-Parametric Methods

- \( K \)-NN method is a special case of non-parametric (or memory-based) methods
  - Non-parametric in the sense that \( f \) are not described by only few parameters
Non-Parametric Methods

- $K$-NN method is a special case of *non-parametric* (or *memory-based*) methods
  - Non-parametric in the sense that $f$ are not described by only few parameters
  - Memory-based in that all data (rather than just parameters) need to be memorized during the training process
Non-Parametric Methods

- $K$-NN method is a special case of *non-parametric* (or *memory-based*) methods
  - Non-parametric in the sense that $f$ are not described by only few parameters
  - Memory-based in that all data (rather than just parameters) need to be memorized during the training process
- $K$-NN is also a *lazy* method since the prediction function $f$ is obtained only before the prediction.
Non-Parametric Methods

- $K$-NN method is a special case of non-parametric (or memory-based) methods
  - Non-parametric in the sense that $f$ are not described by only few parameters
  - Memory-based in that all data (rather than just parameters) need to be memorized during the training process
- $K$-NN is also a lazy method since the prediction function $f$ is obtained only before the prediction
  - Motivates the development of other local models
Pros & Cons

- Pros:
  - Almost no assumption on $f$ other than smoothness
    - High capacity/complexity
    - High accuracy given a large training set

- Cons:
  - Storage demanding
  - Sensitive to outliers
  - Sensitive to irrelevant data features (vs. decision trees)
  - Needs to deal with missing data (e.g., special distances)
  - Computationally expensive: $O(ND)$ time for making each prediction
    - Can speed up with index and/or approximation
Pros & Cons

- **Pros:**
  - Almost no assumption on $f$ other than *smoothness*
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
Pros & Cons

- Pros:
  - Almost no assumption on $f$ other than *smoothness*
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems

- Cons:
  - Storage demanding
  - Sensitive to outliers
  - Sensitive to irrelevant data features (vs. decision trees)
  - Needs to deal with missing data (e.g., special distances)
  - Computationally expensive: $O(ND)$ time for making each prediction
  - Can speed up with index and/or approximation
Pros & Cons

Pros:
- Almost no assumption on $f$ other than *smoothness*
  - High capacity/complexity
  - High accuracy given a large training set
- Supports online training (by simply memorizing)
- Readily extensible to multi-class and regression problems

Cons:
- Storage demanding
Pros & Cons

Pros:

- Almost no assumption on $f$ other than smoothness
  - High capacity/complexity
  - High accuracy given a large training set
- Supports online training (by simply memorizing)
- Readily extensible to multi-class and regression problems

Cons:

- Storage demanding
- Sensitive to outliers
Pros & Cons

Pros:
- Almost no assumption on $f$ other than smoothness
  - High capacity/complexity
  - High accuracy given a large training set
- Supports online training (by simply memorizing)
- Readily extensible to multi-class and regression problems

Cons:
- Storage demanding
- Sensitive to outliers
- Sensitive to irrelevant data features (vs. decision trees)
Pros & Cons

- **Pros:**
  - Almost no assumption on $f$ other than \textit{smoothness}
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems

- **Cons:**
  - Storage demanding
  - Sensitive to outliers
  - Sensitive to irrelevant data features (vs. decision trees)
  - Needs to deal with missing data (e.g., special distances)
Pros & Cons

Pros:

- Almost no assumption on $f$ other than smoothness
  - High capacity/complexity
  - High accuracy given a large training set
- Supports online training (by simply memorizing)
- Readily extensible to multi-class and regression problems

Cons:

- Storage demanding
- Sensitive to outliers
- Sensitive to irrelevant data features (vs. decision trees)
- Needs to deal with missing data (e.g., special distances)
- Computationally expensive: $O(ND)$ time for making each prediction
  - Can speed up with index and/or approximation
Outline

1 Non-Parametric Methods
   - K-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Parzen Windows and Kernels

- Binary KNN classifier:

\[ f(\mathbf{x}) = \text{sign}\left( \sum_{i: \mathbf{x}^{(i)} \in \text{KNN}(\mathbf{x})} y^{(i)} \right) \]

- The “radius” of voter boundary depends on the input \( \mathbf{x} \)
Parzen Windows and Kernels

- Binary KNN classifier:

\[ f(x) = \text{sign} \left( \sum_{i: x(i) \in \text{KNN}(x)} y(i) \right) \]

- The “radius” of voter boundary depends on the input \( x \)

- We can instead use the *Parzen window* with a fixed radius:

\[ f(x) = \text{sign} \left( \sum_i y(i) 1(x(i); \| x(i) - x \| \leq R) \right) \]
Parzen Windows and Kernels

- Binary KNN classifier:

\[ f(x) = \text{sign} \left( \sum_{i : x(i) \in \text{KNN}(x)} y(i) \right) \]

- The “radius” of voter boundary depends on the input \( x \)

- We can instead use the \textit{Parzen window} with a fixed radius:

\[ f(x) = \text{sign} \left( \sum_i y(i) \mathbf{1}(x(i); \|x(i) - x\| \leq R) \right) \]

- Parzen windows also replace the hard boundary with a soft one:

\[ f(x) = \text{sign} \left( \sum_i y(i) k(x(i), x) \right) \]

- \( k(x(i), x) \) is a \textit{radial basis function (RBF) kernel} whose value decreases along space radiating outward from \( x \)
Parzen Windows and Kernels

- Binary KNN classifier:
  \[ f(x) = \text{sign} \left( \sum_{i:x^{(i)} \in \text{KNN}(x)} y^{(i)} \right) \]

- The “radius” of voter boundary depends on the input \( x \)

- We can instead use the Parzen window with a fixed radius:
  \[ f(x) = \text{sign} \left( \sum_i y^{(i)} 1(x^{(i)}; \|x^{(i)} - x\| \leq R) \right) \]

- Parzen windows also replace the hard boundary with a soft one:
  \[ f(x) = \text{sign} \left( \sum_i y^{(i)} k(x^{(i)}, x) \right) \]

- \( k(x^{(i)}, x) \) is a radial basis function (RBF) kernel whose value decreases along space radiating outward from \( x \)
Common RBF Kernels

- How to act like soft $K$-NN?
Common RBF Kernels

- How to act like soft $K$-NN?
- Gaussian RBF kernel:

$$k(x^{(i)}, x) = \mathcal{N}(x^{(i)} - x; 0, \sigma^2 I)$$
Common RBF Kernels

- How to act like soft \( K \)-NN?
- Gaussian RBF kernel:
  \[
  k(x^{(i)}, x) = \mathcal{N}(x^{(i)} - x; 0, \sigma^2 I)
  \]
- Or simply
  \[
  k(x^{(i)}, x) = \exp\left(-\gamma \|x^{(i)} - x\|^2\right)
  \]
- \(\gamma \geq 0 \) (or \(\sigma^2\)) is a hyperparameter controlling the smoothness of \(f\)
Outline

1 Non-Parametric Methods
   • K-NN
   • Parzen Windows
   • Local Models

2 Support Vector Machines
   • SVC
   • Slacks
   • Nonlinear SVC
   • Dual Problem
   • Kernel Trick
Locally Weighted Linear Regression

- In addition to the majority voting and average, we can define *local models* for lazy predictions.
In addition to the majority voting and average, we can define *local models* for lazy predictions.

E.g., in (eager) linear regression, we find $w \in \mathbb{R}^{D+1}$ that minimizes SSE:

$$\arg \min_w \sum_i (y^{(i)} - w^T x^{(i)})^2$$

Local model: to find $w$ minimizing *SSE local to the point $x$ we want to predict*:

$$\arg \min_w \sum_i k(x^{(i)}, x)(y^{(i)} - w^T x^{(i)})^2$$

$k(\cdot, \cdot) \in \mathbb{R}$ is an RBF kernel.
Outline

1 Non-Parametric Methods
   - K-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Kernel Machines

- **Kernel machines**:  
  \[ f(x) = \sum_{i=1}^{N} c_i k(x^{(i)}, x) + c_0 \]

- For example:
  - Parzen windows: \( c_i = y^{(i)} \) and \( c_0 = 0 \)
  - Locally weighted linear regression: \( c_i = (y^{(i)} - w^T x^{(i)})^2 \) and \( c_0 = 0 \)
Kernel Machines

- **Kernel machines:**

\[
 f(x) = \sum_{i=1}^{N} c_i k(x^{(i)}, x) + c_0
\]

- For example:
  - Parzen windows: \( c_i = y^{(i)} \) and \( c_0 = 0 \)
  - Locally weighted linear regression: \( c_i = (y^{(i)} - w^T x^{(i)})^2 \) and \( c_0 = 0 \)
- The variable \( c \in \mathbb{R}^N \) can be learned in either an eager or lazy manner
- Pros: complex, but highly accurate if regularized well
Sparse Kernel Machines

- To make a prediction, we need to store all examples
- May be infeasible due to
  - Large dataset ($N$)
  - Time limit
  - Space limit
Sparse Kernel Machines

- To make a prediction, we need to store all examples
- May be infeasible due to
  - Large dataset \((N)\)
  - Time limit
  - Space limit
- Can we make \(c\) sparse?
  - I.e., to make \(c_i \neq 0\) for only a small fraction of examples called support vectors
- How?
Outline

1 Non-Parametric Methods
   - K-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Separating Hyperplane I

- Model: $\mathcal{F} = \{f : f(x; w, b) = w^\top x + b\}$
  - A collection of hyperplanes
- Prediction: $\hat{y} = \text{sign}(f(x))$
Separating Hyperplane I

- Model: $\mathbb{F} = \{ f : f(x; w, b) = w^\top x + b \}$
  - A collection of hyperplanes
- Prediction: $\hat{y} = \text{sign}(f(x))$
- Training: to find $w$ and $b$ such that

  $\begin{align*}
  w^\top x^{(i)} + b &\geq 0, \quad \text{if } y^{(i)} = 1 \\
  w^\top x^{(i)} + b &\leq 0, \quad \text{if } y^{(i)} = -1
  \end{align*}$
Separating Hyperplane I

- Model: \( \mathcal{F} = \{ f : f(x; w, b) = w^\top x + b \} \)
  - A collection of hyperplanes
- Prediction: \( \hat{y} = \text{sign}(f(x)) \)
- Training: to find \( w \) and \( b \) such that

\[
\begin{align*}
    w^\top x^{(i)} + b &\geq 0, \quad \text{if } y^{(i)} = 1 \\
    w^\top x^{(i)} + b &\leq 0, \quad \text{if } y^{(i)} = -1
\end{align*}
\]

or simply

\[
y^{(i)}(w^\top x^{(i)} + b) \geq 0
\]
Separating Hyperplane II

- There are many feasible $w$’s and $b$’s when the classes are linearly separable.
- Which hyperplane is the best?
Support Vector Classification

- **Support vector classifier (SVC)** picks one with **largest margin**:
  - \( y^{(i)}(w^\top x^{(i)} + b) \geq a \) for all \( i \)
  - Margin: \( 2a/\|w\| \) [Homework]
Support Vector Classification

- **Support vector classifier** (SVC) picks one with *largest margin*:
  - $y^{(i)}(w^T x^{(i)} + b) \geq a$ for all $i$
  - Margin: $2a/\|w\|$ [Homework]

With loss of generality, we let $a = 1$ and solve the problem:

$$\arg \min_{w, b} \frac{1}{2}\|w\|^2$$

subject to $y^{(i)}(w^T x^{(i)} + b) \geq 1, \forall i$
Outline

1. Non-Parametric Methods
   - $K$-NN
   - Parzen Windows
   - Local Models

2. Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Overlapping Classes

- In practice, classes may be overlapping
  - Due to, e.g., noises or outliers
Overlapping Classes

- In practice, classes may be overlapping
  - Due to, e.g., noises or outliers

The problem

\[
\arg \min_{w,b} \frac{1}{2} \|w\|^2
\]

subject to \(y^{(i)}(w^\top x^{(i)} + b) \geq 1, \forall i\)

has no solution in this case. How to fix this?
Slacks

- SVC tolerates *slacks* that fall outside of the regions they ought to be
- Problem:

\[
\begin{aligned}
\arg \min_{w, b, \xi} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i \\
\text{s.t.} & \quad y^{(i)} (w^\top x^{(i)} + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i
\end{aligned}
\]

- Favors large margin but also fewer slacks
Hyperparameter $C$

$$\arg\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \xi_i$$

- The hyperparameter $C$ controls the tradeoff between
  - Maximizing margin
  - Minimizing number of slacks
Hyperparameter $C$

$$\arg\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i$$

- The hyperparameter $C$ controls the tradeoff between
  - Maximizing margin
  - Minimizing number of slacks

- Provides a geometric explanation to the weight decay
Outline

1 Non-Parametric Methods
   - K-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Nonlinearly Separable Classes

- In practice, classes may be nonlinearly separable
Nonlinearly Separable Classes

- In practice, classes may be nonlinearly separable

- SVC (with slacks) gives “bad” hyperplanes due to underfitting
- How to make it nonlinear?
Feature Augmentation

- Recall that in polynomial regression, we augment data features to make a linear regressor nonlinear
Feature Augmentation

- Recall that in polynomial regression, we augment data features to make a linear regressor nonlinear.
- We can define a function \( \Phi(\cdot) \) that maps each data point to a high dimensional space:

\[
\arg\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\
\text{s.t. } y^{(i)}(w^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i
\]
Outline

1 Non-Parametric Methods
   - K-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Time Complexity

- Nonlinear SVC:

\[
\text{arg min}_{w, b, \xi} \frac{1}{2}||w||^2 + C \sum_{i} \xi_i \\
\text{subject to } y^{(i)}(w^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i
\]

- The higher augmented feature dimension, the more variables in \(w\) to solve
Time Complexity

- Nonlinear SVC:

\[
\arg\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i
\]

subject to \(y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i\) and \(\xi_i \geq 0, \forall i\)

- The higher augmented feature dimension, the more variables in \(\mathbf{w}\) to solve

- Can we solve \(\mathbf{w}\) in time complexity that is independent with the mapped dimension?
Dual Problem

- Primal problem:

\[
\begin{align*}
\text{arg min}_{w, b, \xi} & \quad \frac{1}{2}\|w\|^2 + C \sum_i \xi_i \\
\text{subject to} & \quad y^{(i)}(w^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i
\end{align*}
\]

- Dual problem:

\[
\begin{align*}
\text{arg max}_{\alpha, \beta} \text{ min}_{w, b, \xi} & \quad L(w, b, \xi, \alpha, \beta) \\
\text{subject to} & \quad \alpha \geq 0, \beta \geq 0
\end{align*}
\]

where \( L(w, b, \xi, \alpha, \beta) = \)

\[
\begin{align*}
\frac{1}{2}\|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i(1 - y^{(i)}(w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)
\end{align*}
\]
Dual Problem

- Primal problem:
  \[
  \arg\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i
  \]
  subject to \( y(i)(w^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0, \forall i \)

- Dual problem:
  \[
  \arg\max_{\alpha,\beta} \min_{w,b,\xi} L(w,b,\xi,\alpha,\beta)
  \]
  subject to \( \alpha \geq 0, \beta \geq 0 \)

where \( L(w,b,\xi,\alpha,\beta) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y(i)(w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i (-\xi_i) \)

- Primal problem is convex, so \textit{strong duality} holds
Solving Dual Problem I

\[ L(w, b, \xi, \alpha, \beta) = \]
\[
\frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y^{(i)}(w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i (-\xi_i)
\]

The inner problem

\[
\min_{w, b, \xi} L(w, b, \xi, \alpha, \beta)
\]

is convex in terms of \( w, b, \) and \( \xi \)

Let’s solve it analytically:
Solving Dual Problem I

- \( L(w, b, \xi, \alpha, \beta) = \frac{1}{2} ||w||^2 + C \sum \xi_i + \sum \alpha_i (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum \beta_i (-\xi_i) \)

- The inner problem

\[
\min_{w, b, \xi} L(w, b, \xi, \alpha, \beta)
\]

is convex in terms of \( w, b, \) and \( \xi \)

- Let’s solve it analytically:

\[
\frac{\partial L}{\partial w} = w - \sum \alpha_i y^{(i)} \Phi(x^{(i)}) = 0 \Rightarrow w = \sum \alpha_i y^{(i)} \Phi(x^{(i)})
\]
Solving Dual Problem I

- \( L(w, b, \xi, \alpha, \beta) = \)
  \[
  \frac{1}{2}\|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i (-\xi_i)
  \]

- The inner problem
  \[
  \min_{w,b,\xi} L(w, b, \xi, \alpha, \beta)
  \]

  is convex in terms of \( w, b, \) and \( \xi \)

- Let’s solve it analytically:
  \[
  \frac{\partial L}{\partial w} = w - \sum_i \alpha_i y^{(i)} \Phi(x^{(i)}) = 0 \Rightarrow w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})
  \]
  \[
  \frac{\partial L}{\partial b} = \sum_i \alpha_i y^{(i)} = 0
  \]
Solving Dual Problem I

- \( L(w, b, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y^{(i)}(w^T \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i (-\xi_i) \)

- The inner problem

\[
\min_{w, b, \xi} L(w, b, \xi, \alpha, \beta)
\]

is convex in terms of \( w, b, \) and \( \xi \)

- Let’s solve it analytically:

\[
\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y^{(i)} \Phi(x^{(i)}) = 0 \Rightarrow w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})
\]

\[
\frac{\partial L}{\partial b} = \sum_i \alpha_i y^{(i)} = 0
\]

\[
\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \Rightarrow \beta_i = C - \alpha_i
\]
Solving Dual Problem II

- \( L(w, b, \xi, \alpha, \beta) = \)
  \[
  \frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i (-\xi_i)
  \]
- Substituting \( w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)}) \) and \( \beta_i = C - \alpha_i \) in \( L(w, b, \xi, \alpha, \beta) \):
  
  \[
  L(w, b, \xi, \alpha, \beta) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)}) - b \sum_i \alpha_i y^{(i)},
  \]
Solving Dual Problem II

- \( L(w, b, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i (-\xi_i) \)

- Substituting \( w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)}) \) and \( \beta_i = C - \alpha_i \) in \( L(w, b, \xi, \alpha, \beta) \):

\[
L(w, b, \xi, \alpha, \beta) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)}) - b \sum_i \alpha_i y^{(i)},
\]

\[
\min_{w, b, \xi, \alpha, \beta} L(w, b, \xi, \alpha, \beta) = \begin{cases} 
\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)}) , & \text{if } \sum_i \alpha_i y^{(i)} = 0, \\
-\infty, & \text{otherwise}
\end{cases}
\]
Solving Dual Problem II

- \( L(w, b, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) + \sum_i \beta_i (-\xi_i) \)
- Substituting \( w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)}) \) and \( \beta_i = C - \alpha_i \) in \( L(w, b, \xi, \alpha, \beta) \):

\[
L(w, b, \xi, \alpha, \beta) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)}) - b \sum_i \alpha_i y^{(i)},
\]

\[
\min_{w,b,\xi} L(w, b, \xi, \alpha, \beta) = \begin{cases} 
\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)}) , & \text{if } \sum \alpha_i y^{(i)} = 0, \\
-\infty , & \text{otherwise}
\end{cases}
\]

- Outer maximization problem:

\[
\arg \max_\alpha 1^\top \alpha - \frac{1}{2} \alpha^\top K \alpha \\
\text{subject to } 0 \leq \alpha \leq C 1 \text{ and } y^\top \alpha = 0
\]

\[
K_{i,j} = y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)})
\]
Solving Dual Problem II

- \( L(w, b, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i + \sum_i \alpha_i (1 - y(i) (w^T \Phi(x(i)) + b) - \xi_i) + \sum_i \beta_i (-\xi_i) \)

- Substituting \( w = \sum_i \alpha_i y(i) \Phi(x(i)) \) and \( \beta_i = C - \alpha_i \) in \( L(w, b, \xi, \alpha, \beta) \):

\[
L(w, b, \xi, \alpha, \beta) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y(i) y(j) \Phi(x(i))^T \Phi(x(j)) - b \sum_i \alpha_i y(i),
\]

\[
\min_{w,b,\xi} L(w, b, \xi, \alpha, \beta) = \begin{cases} 
\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y(i) y(j) \Phi(x(i))^T \Phi(x(j)), & \text{if } \sum_i \alpha_i y(i) = 0, \\
-\infty, & \text{otherwise}
\end{cases}
\]

- Outer maximization problem:

\[
\arg \max_\alpha 1^T \alpha - \frac{1}{2} \alpha^T K \alpha \\
\text{subject to } 0 \leq \alpha \leq C1 \text{ and } y^T \alpha = 0
\]

- \( K_{i,j} = y(i) y(j) \Phi(x(i))^T \Phi(x(j)) \)

- \( \beta_i = C - \alpha_i \geq 0 \) implies \( \alpha_i \leq C \)
Solving Dual Problem II

- Dual minimization problem of SVC:

\[
\arg \min_{\alpha} \frac{1}{2}\alpha^\top K\alpha - 1^\top \alpha
\]
subject to \(0 \leq \alpha \leq C1\) and \(y^\top \alpha = 0\)

- Number of variables to solve?
Solving Dual Problem II

- Dual minimization problem of SVC:

\[
\arg\min_\alpha \frac{1}{2} \alpha^\top K\alpha - 1^\top \alpha \\
\text{subject to } 0 \leq \alpha \leq C1 \text{ and } y^\top \alpha = 0
\]

- Number of variables to solve? \( N \) instead of augmented feature dimension
Solving Dual Problem II

- Dual minimization problem of SVC:

  \[
  \text{arg min}_\alpha \frac{1}{2} \alpha^\top K\alpha - 1^\top \alpha
  \]

  subject to \( 0 \leq \alpha \leq C1 \) and \( y^\top \alpha = 0 \)

- Number of variables to solve? \( N \) instead of augmented feature dimension

- In practice, this problem is solved by specialized solvers such as the sequential minimal optimization (SMO) [3]

  - As \( K \) is usually ill-conditioned
Making Predictions

- Prediction: \( \hat{y} = \text{sign}(f(x)) = \text{sign}(w^T x + b) \)
Making Predictions

- Prediction: $\hat{y} = \text{sign}(f(x)) = \text{sign}(w^\top x + b)$
- We have $w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})$
- How to obtain $b$?

By the complementary slackness of KKT conditions, we have:

$$a_i (y^{(i)} (w^\top F(x^{(i)})) + b) x_i = 0$$

For any $x^{(i)}$ having $0 < a_i < C$, we have $b_i = C a_i > 0$.

\[ x_i \]

In practice, we usually take the average over all $x^{(i)}$'s having $0 < a_i < C$ to avoid numeric error.
Making Predictions

- Prediction: \( \hat{y} = \text{sign}(f(x)) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b) \)
- We have \( \mathbf{w} = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)}) \)
- How to obtain \( b \)?
- By the complementary slackness of KKT conditions, we have:
  \[
  \alpha_i (1 - y^{(i)} (\mathbf{w}^\top \Phi(x^{(i)}) + b) - \xi_i) = 0 \quad \text{and} \quad \beta_i (-\xi_i) = 0
  \]
Making Predictions

- Prediction: \( \hat{y} = \text{sign}(f(x)) = \text{sign}(w^\top x + b) \)
- We have \( w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)}) \)
- How to obtain \( b \)?
- By the complementary slackness of KKT conditions, we have:
  \[
  \alpha_i (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) = 0 \quad \text{and} \quad \beta_i (-\xi_i) = 0
  \]
- For any \( x^{(i)} \) having \( 0 < \alpha_i < C \), we have
  \[
  \beta_i = C - \alpha_i > 0 \Rightarrow \xi_i = 0,
  \]
  \[
  (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) = 0 \Rightarrow b = y^{(i)} - w^\top \Phi(x^{(i)})
  \]
Making Predictions

- Prediction: $\hat{y} = \text{sign}(f(x)) = \text{sign}(w^\top x + b)$
- We have $w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})$
- How to obtain $b$?
- By the complementary slackness of KKT conditions, we have:
  \[ \alpha_i (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) = 0 \text{ and } \beta_i (-\xi_i) = 0 \]
- For any $x^{(i)}$ having $0 < \alpha_i < C$, we have
  \[ \beta_i = C - \alpha_i > 0 \Rightarrow \xi_i = 0, \]
  \[ (1 - y^{(i)} (w^\top \Phi(x^{(i)}) + b) - \xi_i) = 0 \Rightarrow b = y^{(i)} - w^\top \Phi(x^{(i)}) \]
- In practice, we usually take the average over all $x^{(i)}$'s having $0 < \alpha_i < C$ to avoid numeric error
Outline

1 Non-Parametric Methods
   - $K$-NN
   - Parzen Windows
   - Local Models

2 Support Vector Machines
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick
Kernel as Inner Product

- We need to evaluate $\Phi(x^{(i)})^\top \Phi(x^{(j)})$ when
- Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)})$
- Making a prediction, where $f(x) = w^\top x + b = \sum \alpha_i y^{(i)} \Phi(x^{(i)})^\top \Phi(x) + b$
Kernel as Inner Product

- We need to evaluate $\Phi(x^{(i)})^\top \Phi(x^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)})$
  - Making a prediction, where $f(x) = w^\top x + b = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})^\top \Phi(x) + b$
- Time complexity?
Kernel as Inner Product

- We need to evaluate $\Phi(x^{(i)})^\top \Phi(x^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)}y^{(j)}\Phi(x^{(i)})^\top \Phi(x^{(j)})$
  - Making a prediction, where $f(x) = w^\top x + b = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})^\top \Phi(x) + b$

- Time complexity?
- If we choose $\Phi$ carefully, we can can evaluate $\Phi(x^{(i)})^\top \Phi(x) = k(x^{(i)}, x)$ efficiently
Kernel as Inner Product

- We need to evaluate $\Phi(x^{(i)})^\top \Phi(x^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)})$
  - Making a prediction, where $f(x) = w^\top x + b = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})^\top \Phi(x) + b$

- Time complexity?
- If we choose $\Phi$ carefully, we can efficiently evaluate $\Phi(x^{(i)})^\top \Phi(x) = k(x^{(i)}, x)$
- Polynomial kernel: $k(a, b) = (a^\top b / \alpha + \beta)^\gamma$
  - E.g., let $\alpha = 1$, $\beta = 1$, $\gamma = 2$ and $a \in \mathbb{R}^2$, then $\Phi(a) = [1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, a_2^2, \sqrt{2}a_1a_2]^\top \in \mathbb{R}^6$
Kernel as Inner Product

- We need to evaluate \( \Phi(x^{(i)})^\top \Phi(x^{(j)}) \) when
  - Solving dual problem of SVC, where \( K_{i,j} = y^{(i)}y^{(j)}\Phi(x^{(i)})^\top \Phi(x^{(j)}) \)
  - Making a prediction, where \( f(x) = w^\top x + b = \sum_i \alpha_i y^{(i)}\Phi(x^{(i)})^\top \Phi(x) + b \)

- Time complexity?

- If we choose \( \Phi \) carefully, we can evaluate \( \Phi(x^{(i)})^\top \Phi(x) = k(x^{(i)}, x) \) efficiently

- Polynomial kernel: \( k(a,b) = (a^\top b / \alpha + \beta)^\gamma \)
  - E.g., let \( \alpha = 1, \beta = 1, \gamma = 2 \) and \( a \in \mathbb{R}^2 \), then
    \[
    \Phi(a) = [1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, a_2^2, \sqrt{2}a_1a_2]^\top \in \mathbb{R}^6
    \]

- Gaussian RBF kernel: \( k(a,b) = \exp(-\gamma \|a-b\|^2) \), \( \gamma \geq 0 \)
  - \( k(a,b) = \exp(-\gamma \|a\|^2 + 2\gamma a^\top b - \gamma \|b\|^2) = \exp(-\gamma \|a\|^2 - \gamma \|b\|^2)(1 + \frac{2\gamma a^\top b}{1!} + \frac{(2\gamma a^\top b)^2}{2!} + \cdots) \)
  - Let \( a \in \mathbb{R}^2 \), then \( \Phi(a) = \exp(-\gamma \|a\|^2)[1, \sqrt{2\gamma} a_1, \sqrt{2\gamma} a_2, \sqrt{2\gamma} a_1^2, \sqrt{2\gamma} a_2^2, 2\sqrt{\gamma} a_1a_2, \cdots]^\top \in \mathbb{R}^\infty \)
Kernel as Inner Product

- We need to evaluate $\Phi(x^{(i)})^\top \Phi(x^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(x^{(i)})^\top \Phi(x^{(j)})$
  - Making a prediction, where $f(x) = w^\top x + b = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})^\top \Phi(x) + b$

- Time complexity?
- If we choose $\Phi$ carefully, we can can evaluate $\Phi(x^{(i)})^\top \Phi(x) = k(x^{(i)}, x)$ efficiently

- Polynomial kernel: $k(a,b) = (a^\top b / \alpha + \beta)^\gamma$
  - E.g., let $\alpha = 1$, $\beta = 1$, $\gamma = 2$ and $a \in \mathbb{R}^2$, then
    \[
    \Phi(a) = [1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, a_2^2, \sqrt{2}a_1 a_2]^\top \in \mathbb{R}^6
    \]

- Gaussian RBF kernel: $k(a,b) = \exp(-\gamma \|a - b\|^2)$, $\gamma \geq 0$
  - $k(a,b) = \exp(-\gamma \|a\|^2 + 2\gamma a^\top b - \gamma \|b\|^2) =$
  - $\exp(-\gamma \|a\|^2 - \gamma \|b\|^2) (1 + \frac{2\gamma a^\top b}{1!} + \frac{(2\gamma a^\top b)^2}{2!} + \cdots)$
  - Let $a \in \mathbb{R}^2$, then $\Phi(a) =$
    \[
    \exp(-\gamma \|a\|^2)[1, \sqrt{\frac{2\gamma}{1!}} a_1, \sqrt{\frac{2\gamma}{1!}} a_2, \sqrt{\frac{2\gamma}{2!}} a_1^2, \sqrt{\frac{2\gamma}{2!}} a_2^2, 2\sqrt{\frac{2\gamma}{2!}} a_1 a_2, \cdots]^\top \in \mathbb{R}^{\infty}
    \]
Kernel Trick

- If we choose $\Phi$ induced by Polynomial or Gaussian RBF kernel, then

$$K_{i,j} = y^{(i)}y^{(j)}k(x^{(i)}, x)$$

takes only $O(D)$ time to evaluate, and

$$f(x) = \sum_i \alpha_i y^{(i)}k(x^{(i)}, x) + b$$

takes $O(ND)$ time
Kernel Trick

- If we choose $\Phi$ induced by Polynomial or Gaussian RBF kernel, then
  \[ K_{i,j} = y^{(i)}y^{(j)}k(x^{(i)}, x) \]
  takes only $O(D)$ time to evaluate, and

  \[ f(x) = \sum_i \alpha_i y^{(i)}k(x^{(i)}, x) + b \]
  takes $O(ND)$ time

- Independent with the augmented feature dimension
Kernel Trick

- If we choose \( \Phi \) induced by Polynomial or Gaussian RBF kernel, then
  \[
  K_{i,j} = y^{(i)}y^{(j)}k(x^{(i)}, x)
  \]
  takes only \( O(D) \) time to evaluate, and
  \[
  f(x) = \sum_{i} \alpha_i y^{(i)}k(x^{(i)}, x) + b
  \]
  takes \( O(ND) \) time
- Independent with the augmented feature dimension
- \( \alpha, \beta, \) and \( \gamma \) are new hyperparameters
Sparse Kernel Machines

- SVC is a kernel machine:
  \[ f(x) = \sum_i \alpha_i y^{(i)} k(x^{(i)}, x) + b \]

- It is surprising that SVC works like $K$-NN in some sense
Sparse Kernel Machines

- SVC is a kernel machine:
  \[ f(x) = \sum_i \alpha_i y^{(i)} k(x^{(i)}, x) + b \]

- It is surprising that SVC works like K-NN in some sense
- However, SVC is a sparse kernel machine
- Only the slacks become the support vectors \((\alpha_i > 0)\)
KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: \( y(i)(w^\top \Phi(x(i)) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
  - Complementary slackness: \( \alpha_i(1 - y(i)(w^\top \Phi(x(i)) + b) - \xi_i) = 0 \) and \( \beta_i(-\xi_i) = 0 \)
KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: \( y(i)(w^\top \Phi(x(i)) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
  - Complementary slackness: \( \alpha_i (1 - y(i)(w^\top \Phi(x(i)) + b) - \xi_i) = 0 \) and \( \beta_i (-\xi_i) = 0 \)

- Depending on the value of \( \alpha_i \), each example \( x(i) \) can be:
KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: \( y^{(i)}(w^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
  - Complementary slackness: \( \alpha_i(1 - y^{(i)}(w^\top \Phi(x^{(i)}) + b) - \xi) = 0 \) and \( \beta_i(-\xi_i) = 0 \)

- Depending on the value of \( \alpha_i \), each example \( x^{(i)} \) can be:
KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: \( y(i)(w^\top \Phi(x(i)) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
  - Complementary slackness: \( \alpha_i(1 - y(i)(w^\top \Phi(x(i)) + b) - \xi) = 0 \) and \( \beta_i(-\xi) = 0 \)

- Depending on the value of \( \alpha_i \), each example \( x(i) \) can be:
  - Non SVs (\( \alpha_i = 0 \)): \( y(i)(w^\top \Phi(x(i)) + b) \geq 1 \) (usually strict)
  - Free SVs (\( 0 < \alpha_i < C \)): \( y(i)(w^\top \Phi(x(i)) + b) = 1 \)
  - Bounded SVs (\( \alpha_i = C \)): \( y(i)(w^\top \Phi(x(i)) + b) \leq 1 \) (usually strict)
KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: \( y^{(i)}(w^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
  - Complementary slackness: \( \alpha_i (1 - y^{(i)}(w^\top \Phi(x^{(i)}) + b) - \xi_i) = 0 \) and \( \beta_i(-\xi_i) = 0 \)

- Depending on the value of \( \alpha_i \), each example \( x^{(i)} \) can be:
  - Non SVs (\( \alpha_i = 0 \)): \( y^{(i)}(w^\top \Phi(x^{(i)}) + b) \geq 1 \) (usually strict)
    - \( 1 - y^{(i)}(w^\top \Phi(x^{(i)}) + b) - \xi_i \leq 0 \)
    - Since \( \beta_i = C - \alpha_i \neq 0 \), we have \( \xi_i = 0 \)
  - Free SVs (\( 0 < \alpha_i < C \)): \( y^{(i)}(w^\top \Phi(x^{(i)}) + b) = 1 \)

- Bounded SVs (\( \alpha_i = C \)): \( y^{(i)}(w^\top \Phi(x^{(i)}) + b) \leq 1 \) (usually strict)
KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: \( y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
  - Complementary slackness: \( \alpha_i (1 - y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) - \xi_i) = 0 \) and \( \beta_i(-\xi_i) = 0 \)

- Depending on the value of \( \alpha_i \), each example \( x^{(i)} \) can be:
  - Non SVs (\( \alpha_i = 0 \)): \( y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) \geq 1 \) (usually strict)
    - \( 1 - y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) - \xi_i \leq 0 \)
    - Since \( \beta_i = C - \alpha_i \neq 0 \), we have \( \xi_i = 0 \)
  - Free SVs (\( 0 < \alpha_i < C \)): \( y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) = 1 \)
    - \( 1 - y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) - \xi_i = 0 \)
    - Since \( \beta_i = C - \alpha_i \neq 0 \), we have \( \xi_i = 0 \)
  - Bounded SVs (\( \alpha_i = C \)): \( y^{(i)}(\mathbf{w}^\top \Phi(x^{(i)}) + b) \leq 1 \) (usually strict)
KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: \( y(i)(w^T \Phi(x(i)) + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
  - Complementary slackness: \( \alpha_i(1 - y(i)(w^T \Phi(x(i)) + b) - \xi_i) = 0 \) and \( \beta_i(-\xi_i) = 0 \)

- Depending on the value of \( \alpha_i \), each example \( x(i) \) can be:
  - Non SVs \( (\alpha_i = 0) \): \( y(i)(w^T \Phi(x(i)) + b) \geq 1 \) (usually strict)
    - \( 1 - y(i)(w^T \Phi(x(i)) + b) - \xi_i \leq 0 \)
    - Since \( \beta_i = C - \alpha_i \neq 0 \), we have \( \xi_i = 0 \)

  - Free SVs \( (0 < \alpha_i < C) \): \( y(i)(w^T \Phi(x(i)) + b) = 1 \)
    - \( 1 - y(i)(w^T \Phi(x(i)) + b) - \xi_i = 0 \)
    - Since \( \beta_i = C - \alpha_i \neq 0 \), we have \( \xi_i = 0 \)

  - Bounded SVs \( (\alpha_i = C) \): \( y(i)(w^T \Phi(x(i)) + b) \leq 1 \) (usually strict)
    - \( 1 - y(i)(w^T \Phi(x(i)) + b) - \xi_i = 0 \)
    - Since \( \beta_i = 0 \), we have \( \xi_i \geq 0 \)
Remarks 1

- Pros of SVC:
  - Global optimality (convex problem)

- Cons:
  - Nonlinear SVC not scalable to large tasks
  - Takes \(O(N^2) \ll O(N^3)\) time to train using SMO in LIBSVM [1]
  - Kernel matrix \(K\) requires \(O(N^2)\) space
  - In practice, we cache only a small portion of \(K\) in memory
  - Sensitive to irrelevant data features (vs. decision trees)
  - Non-trivial hyperparameter tuning
    - The effect of a \((C, \gamma)\) combination is unknown in advance
      - Usually done by grid search
  - Usually wrapped by the 1-vs-1 technique for multi-class classification
Remarks 1

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
Remarks I

- **Pros of SVC:**
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set

- **Cons:**
  - Nonlinear SVC not scalable to large tasks
  - Time to train using SMO in LIBSVM \([\mathcal{O}(N^3)]\)
    - On the other hand, linear SVC takes \([\mathcal{O}(ND)]\) time
  - Kernel matrix \([K]\) requires \([\mathcal{O}(N^2)]\) space
    - In practice, we cache only a small portion of \([K]\) in memory
  - Sensitive to irrelevant data features (vs. decision trees)
  - Non-trivial hyperparameter tuning
    - The effect of \((C, g)\) combination is unknown in advance
      - Usually done by grid search
  - Separate only 2 classes
    - Usually wrapped by the 1-vs-1 technique for multi-class classification
Remarks I

- **Pros of SVC:**
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set

- **Cons:**
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
Remarks I

- **Pros of SVC:**
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set

- **Cons:**
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
  - Kernel matrix $K$ requires $O(N^2)$ space
    - In practice, we cache only a small portion of $K$ in memory
Remarks I

- **Pros of SVC:**
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set

- **Cons:**
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
  - Kernel matrix $K$ requires $O(N^2)$ space
    - In practice, we cache only a small portion of $K$ in memory
  - Sensitive to irrelevant data features (vs. decision trees)
Remarks I

- **Pros of SVC:**
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set

- **Cons:**
  - Nonlinear SVC not scalable to large tasks
    - Takes \( O(N^2) \sim O(N^3) \) time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes \( O(ND) \) time
  - Kernel matrix \( K \) requires \( O(N^2) \) space
    - In practice, we cache only a small portion of \( K \) in memory
  - Sensitive to irrelevant data features (vs. decision trees)
  - Non-trivial hyperparameter tuning
    - The effect of a \( (C, \gamma) \) combination is unknown in advance
    - Usually done by **grid search**
Remarks I

- **Pros of SVC:**
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set

- **Cons:**
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
  - Kernel matrix $K$ requires $O(N^2)$ space
    - In practice, we cache only a small portion of $K$ in memory
  - Sensitive to irrelevant data features (vs. decision trees)
  - Non-trivial hyperparameter tuning
    - The effect of a $(C, \gamma)$ combination is unknown in advance
    - Usually done by *grid search*
  - Separate only 2 classes
    - Usually wrapped by the 1-vs-1 technique for multi-class classification
Remarks II

- Does nonlinear SVC always perform better than linear SVC?
Remarks II

- Does nonlinear SVC always perform better than linear SVC? \textit{No}
- Choose linear SVC (e.g., LIBLINEAR [2]) when
  - \( N \) is large (since nonlinear SVC does not scale), or
  - \( D \) is large (since classes may already be linearly separable)
[1] Chih-Chung Chang and Chih-Jen Lin.
Libsvm: a library for support vector machines.

Liblinear: A library for large linear classification.

Sequential minimal optimization: A fast algorithm for training support vector machines.
1998.